React recoil Asynchronous (비동기처리)

sbjang123456 | 2021. 6. 11. 21:56

recoil 을 사용하면서 가장 마음에 들었던 부분은 비동기 처리가 별도 라이브러리 없이 recoil 만으로 가능하다는 것이다.

recoil 의 selector 를 활용하면 쉽게 비동기 데이터를 가져올 수 있다.

이를 위해 recoil state 를 작성한다. (./src/states/board.js)

```
1  import { atom, selector } from 'recoil';
2  import axios from 'axios';
3
4  export const boardSearchState = atom({
5      key: 'boardSearchState',
6      default: {
7          writer: '',
8          title: '',
9          content: '',
10     }
11 });
12
13 export const forceReloadBoardListState = atom({
14     key: 'forceReloadBoardListState',
15     default: 0
16 });
17
18 export const boardListSelector = selector({
19     key: 'boardListSelector',
20     get: async ({ get }) => {
21         get(forceReloadBoardListState);
22         const searchParams = get(boardSearchState);
23
24         const { data } = await axios.get('/api/v1/board', {
25             params: searchParams
26         });
27
28         return data;
29     },
30     set: ({ set }) => {
31         set(forceReloadBoardListState, Math.random());
32     }
33 });
```


- Line 4~11 : board 검색 조건에 대한 atom 객체

- Line 13~16 : board 를 리로드하기 위한 atom 객체
- Line 18~33 : board list 를 가져오기 위한 비동기 selector 객체
- Line 21 : 리로드를 위해 selector 에서 구독
- Line 22 : 검색조건 구독
- Line 24~26 : 비동기 요청
- Line 30~32 : selector 의 set 함수 이지만 내부에서 forceReloadBoardListState 객체의 상태 값을 랜덤 숫자로 변경. forceReloadBoardListState 를 구독하고 있는 selector 를 강제로 리로드 하기 위함.

비동기 selector 를 구독하는 컴포넌트 생성 (./src/components/Board.jsx)

```
1  import React, { Suspense } from 'react';
2  import { useRecoilValue } from 'recoil';
3  import { boardListSelector } from 'states/board';
4
5  const Board = () => {
6      const boardList = useRecoilValue(boardListSelector);
7
8      return (
9          <Suspense fallback={<div>Loading...</div>}>
10             <table>
11                 <tbody>
12                     {boardList.map((row, idx) => (
13                         <tr key={idx}>
14                             <td>{row.number}</td>
15                             <td>{row.title}</td>
16                             <td>{row.writer}</td>
17                             <td>{row.createdAt}</td>
18                         </tr>
19                     ))}
20                 </tbody>
21             </table>
22         </Suspense>
23     );
24 }
25
26 export default Board;
```

- Line 6 : selector 를 useRecoilValue 함수를 통해 구독
- Line 9 : 비동기 selector 의 값을 사용하려면 React 에서 제공하는 Suspense 로 묶어야 함. 이 때, fallback 프로퍼티를 통해 아직 값을 가져오지 못했을 때 렌더링 할 컴포넌트를 제공해야 한다.

하지만 나는 Suspense 로 묶으면 화면이 렌더링 될 때, fallback 이 화면을 가려버려서 화면전환이 매끄럽지 못한것 같다는 느낌을 받았다.

그래서 나는 recoil 에서 제공하는 useRecoilValueLoadable 과 react 의 useMemo 를 사용해서 렌더링 하였다.

```
1  import React, { useMemo } from 'react';
2  import { useRecoilValueLoadable } from 'recoil';
3  import { boardListSelector } from 'states/board';
4
```

```
5  const Board = () => {
6      const boardList = useRecoilValueLoadable(boardListSelector);
7
8      const rows = useMemo(() => {
9          return boardList?.state === 'hasValue' ? boardList?.contents : []
10     }, [boardList]);
11
12     return (
13         <table>
14             <tbody>
15                 {rows.map((row, idx) => (
16                     <tr key={idx}>
17                         <td>{row.number}</td>
18                         <td>{row.title}</td>
19                         <td>{row.writer}</td>
20                         <td>{row.createdAt}</td>
21                     </tr>
22                 ))}
23             </tbody>
24         </table>
25     );
26 }
27
28 export default Board;
```

위 소스는 편의상 작성하였지만 상황에 따라 에러처리나 로딩 처리를 해주는 것이 좋을 것 같다.

> ※ **Loadable 객체는 state 와 contents 프로퍼티를 가지고 있다.**
> **state** : atom 이나 selector 의 상태를 보여준다. (hasValue, hasError, loading)
> **contents** : atom 이나 selector 의 값이며, 상태에 따라 다른 값을 갖고 있다.
> (hasValue - value , hasError - Error , loading - Promise)

- Line 6 : useRecoilValueLoadable 을 통해 selector를 구독
- Line 8~10 : react 의 useMemo Hook 을 통해 해당 selector 가 값이 있으면 데이터를 리턴하고, 아직 갖고 오지 못했으면 빈 배열([]) 을 리턴

또한, 목록 조회에 리로드 기능이 필요하다면 전에 만들었던 selector 의 set 함수를 그대로 호출하면 된다.

```
1  ...
2  import { useRecoilStateLoadable } from 'recoil'
3  ...
4
5  const [boardList, reload] = useRecoilStateLoadable(boardListSelector);
6
7  const handleClickReload = () => {
8        reload();
9  };
10
11 ...
12 생략
13 ...
```